

On Multistage Learning a Hidden Hypergraph

A. G. D'yachkov, I.V. Vorobyev, N.A. Polyanskii and V.Yu. Shchukin

Lomonosov Moscow State University, Moscow, Russia

Email: agd-msu@yandex.ru, vorobyev.i.v@yandex.ru, nikitapolyansky@gmail.com, vpike@mail.ru

Abstract—Learning a hidden hypergraph is a natural generalization of the classical group testing problem that consists in detecting unknown hypergraph $H_{un} = H(V, E)$ by carrying out edge-detecting tests. In the given paper we focus our attention only on a specific family $\mathcal{F}(t, s, \ell)$ of localized hypergraphs for which the total number of vertices $|V| = t$, the number of edges $|E| \leq s$, $s \ll t$, and the cardinality of any edge $|e| \leq \ell$, $\ell \ll t$. Our goal is to identify all edges of $H_{un} \in \mathcal{F}(t, s, \ell)$ by using the minimal number of tests. We develop an adaptive algorithm that matches the information theory bound, i.e., the total number of tests of the algorithm in the worst case is at most $s\ell \log_2 t(1 + o(1))$. We also discuss a probabilistic generalization of the problem.

Keywords: Group testing problem, learning hidden hypergraph, capacity, asymptotic rate, cover-free code

I. INTRODUCTION

A. Notations and Definitions

Let $|A|$ denote the size of a set A , \triangleq denotes the equality by definition and $[N] \triangleq \{1, 2, \dots, N\}$ is the set of integers from 1 to N . A *hypergraph* is the pair $H \triangleq H(V, E)$ such that $E \subset 2^V \setminus \emptyset$, where V is the set of vertices and

$$E \triangleq \{(\mathbf{e}_1, \dots, \mathbf{e}_s) : \mathbf{e}_i \subset V, i \in [s]\}$$

is an s -set of edges. A set $S \subset V$ is called an *independent* set of H if it does not contain entire edges of H . We denote by $\dim(H)$ the cardinality of the largest edge, i.e., $\dim(H) = \max_{i \in [s]} |\mathbf{e}_i|$.

B. Statement of the problem

The problem of learning a hidden hypergraph is described [8] as follows. Suppose there is an unknown (hidden) hypergraph $H_{un} = H(V, E)$ whose edges are not known to us, but we know that the unknown hypergraph H_{un} belongs to the known family \mathcal{F} of hypergraphs having certain specific structure (e.g., \mathcal{F} consists of all Hamiltonian cycles on V). Our goal is to identify all edges of E by carrying out the minimal number N of *edge-detecting queries* $Q(S)$, where $S \subseteq V$: $Q(S) = 0$ if S is independent of H_{un} , and $Q(S) = 1$ otherwise.

In the given paper we focus our attention only on the family $\mathcal{F}(t, s, \ell)$ of hypergraphs introduced in the abstract, namely: given integers s, ℓ and t , such that $s + \ell < t$, the set of vertices $V \triangleq [t]$ and the family $\mathcal{F}(t, s, \ell)$, consists of all hypergraphs $H(V, E)$ such that $\dim(H) \leq \ell$ and $|E| \leq s$. Suppose we know that the hypergraph H_{un} belongs to the family $\mathcal{F}(t, s, \ell)$. An algorithm is said to be an $\mathcal{F}(t, s, \ell)$ -*searching* algorithm if it finds H_{un} , i.e. there exists only one hypergraph from $\mathcal{F}(t, s, \ell)$ that fits all answers to the queries.

One of the most important aspects of any searching strategy is its adaptiveness. An algorithm is *non-adaptive* if all queries are carried out in parallel. An algorithm is *adaptive* if the later queries may depend on the answers to earlier queries.

By $N^{na}(t, s, \ell)$ ($N^a(t, s, \ell)$) denote the minimal number of queries in an $\mathcal{F}(t, s, \ell)$ -searching non-adaptive (adaptive) algorithms. Introduce the *asymptotic rate* of $\mathcal{F}(t, s, \ell)$ -searching non-adaptive algorithms:

$$R^{na}(s, \ell) \triangleq \overline{\lim}_{t \rightarrow \infty} \frac{\log_2 t}{N^{na}(t, s, \ell)},$$

In similar way we define the asymptotic rate $R^a(s, \ell)$ of $\mathcal{F}(t, s, \ell)$ -searching adaptive algorithms.

The rest of the paper is organized as follows. In Sect. II, we discuss previously known results and remind the concept of cover-free codes which is close to the subject. In Sect. III, we present the main result of the paper and provide the deterministic adaptive algorithm that matches the information theory bound. Finally, in Sect. IV we discuss a probabilistic generalization of the problem of learning a hidden hypergraph.

II. PREVIOUS RESULTS

For the particular case $\ell = 1$, the above definitions were already introduced to describe the model called *designing screening experiments*. It is a classical group testing problem. We refer the reader to the monograph [10] for a survey on group testing and its applications. It is quite clear (e.g., see [10]) that an $\mathcal{F}(t, s, 1)$ -searching adaptive algorithm can achieve the information theory bound, i.e., $N(t, s, 1) = s \log_2 t(1 + o(1))$ as $t \rightarrow \infty$. Therefore, $R^a(s, 1) = 1/s$.

If $\ell = 2$, then we deal with learning a hidden graph. One important application area for such problem is bioinformatics [9], more specifically, chemical reactions and genome sequencing. Alon et al. [7], and Alon and Asodi [6] give lower and upper bounds on the minimal number of tests for non-adaptive searching algorithms for certain families of graphs, such as stars, cliques, matchings. In [9], Boevel et al. study the problem of reconstructing a Hamiltonian cycle. In [5], Angluin et al. give a suboptimal $\mathcal{F}(t, s, 2)$ -searching adaptive algorithm. More precisely, they prove $R^a(s, 2) \geq 1/(12s)$.

For the general case of parameters s and ℓ , Abasi et al. have recently provided [11] a suboptimal $\mathcal{F}(t, s, \ell)$ -searching adaptive algorithm. In particular, from their proofs it follows $R^a(s, \ell) \geq 1/(2s\ell)$. This bound differs up to the constant factor from the information theory upper bound $R^a(s, \ell) \leq 1/(s\ell)$. In Sect. III we improve the lower bound and provide an $\mathcal{F}(t, s, \ell)$ -searching adaptive algorithm which is optimal in terms of the asymptotic rate.

A. Cover-Free Codes

A binary $N \times t$ -matrix

$$X = \|x_i(j)\|, \quad x_i(j) = 0, 1, \quad i \in [N], \quad j \in [t] \quad (1)$$

is called a *code of length N and size t* . By x_i and $x(j)$ we denote the i -th row and the j -th column of the code X , respectively.

Before we give the well-known definition of cover-free codes, note that any $\mathcal{F}(t, s, \ell)$ -searching non-adaptive algorithm consisting of N queries can be represented by a binary $N \times t$ matrix X such that each test corresponds to the row, and each vertex stands for the column. We put $x_i(j) = 1$ if the j -th vertex is included to the i -th test; otherwise, $x_i(j) = 0$.

Definition 1. [1]. A code X is called a *cover-free (s, ℓ) -code* (briefly, *CF (s, ℓ) -code*) if for any two non-intersecting sets $S, \mathcal{L} \subset [t]$, $|S| = s$, $|\mathcal{L}| = \ell$, $S \cap \mathcal{L} = \emptyset$, there exists a row x_i , $i \in [N]$, for which

$$\begin{aligned} x_i(j) &= 0 \text{ for any } j \in S, \\ x_i(k) &= 1 \text{ for any } k \in \mathcal{L}. \end{aligned} \quad (2)$$

Taking into account the evident symmetry over s and ℓ , we introduce $N_{cf}(t, s, \ell) = N_{cf}(t, \ell, s)$ - the minimal length of CF (s, ℓ) -codes of size t and define the *asymptotic rate* of CF (s, ℓ) -codes:

$$R_{cf}(s, \ell) = R_{cf}(\ell, s) \triangleq \overline{\lim}_{t \rightarrow \infty} \frac{\log_2 t}{N_{cf}(t, s, \ell)}. \quad (3)$$

In [2], Dyachkov et al. show that any CF (s, ℓ) -code represents a $\mathcal{F}(t, s, \ell)$ -searching non-adaptive algorithm, while any $\mathcal{F}(t, s, \ell)$ -searching non-adaptive algorithm corresponds to both a CF $(s, \ell - 1)$ -code and CF $(s - 1, \ell)$ -code. The best presently known upper and lower bounds on $R(s, \ell)$ of CF (s, ℓ) -codes were presented in [2], [3]. If $\ell \geq 1$ is fixed and $s \rightarrow \infty$, then these bounds lead to the following asymptotic equality:

$$\begin{aligned} \frac{(\ell + 1)^{\ell+1}}{2e^{\ell-1}} \frac{\log_2 s}{s^{\ell+1}} (1 + o(1)) &\geq R^{na}(s, \ell) \\ &\simeq R_{cf}(s, \ell) \geq \frac{\ell^\ell}{e^\ell} \frac{\log_2 e}{s^{\ell+1}} (1 + o(1)). \end{aligned} \quad (4)$$

III. ADAPTIVE LEARNING A HIDDEN HYPERGRAPH

By a counting argument, the lower bound is true.

Theorem 1. Any $\mathcal{F}(t, s, \ell)$ -searching algorithm has at least $s\ell \log_2 t(1 + o(1))$ edge-detecting queries. In other words, the rate $R^a(s, \ell) \leq 1/(s\ell)$.

Proof of Theorem 1.

Let N be the minimal number of tests in the worst case among all adaptive $\mathcal{F}(t, s, \ell)$ -searching algorithms. Then we have

$$2^N \geq |\mathcal{F}(t, s, \ell)|.$$

This inequity along with the asymptotic equality

$$|\mathcal{F}(t, s, \ell)| = \frac{t^{s\ell}}{(\ell!)^s s!} (1 + o(1)), \quad t \rightarrow \infty,$$

leads to

$$\frac{\log_2 t}{N} \leq \frac{1}{s\ell} + o(1), \quad t \rightarrow \infty,$$

Therefore, we complete the proof. \square

The key result of this paper is given as follows.

Theorem 2. There exists an adaptive $\mathcal{F}(t, s, \ell)$ -searching algorithm which has at most $s\ell \log_2 t(1 + o(1))$ edge-detecting queries. In other words, the rate $R^a(s, \ell) = 1/(s\ell)$.

In order to prove Theorem 2 we provide a deterministic $\mathcal{F}(t, s, \ell)$ -searching adaptive algorithm.

Proof of Theorem 2.

Let $H_{un} = H(V, E)$ be a hidden hypergraph from the family $\mathcal{F}(t, s, \ell)$. We will call a vertex $v \in V$ *active*, if there exists at least one edge $e \in E$ such that $v \in e$. By $F, F \subset V$, denote the set of already found active vertices. By $E', E' \subset E$, denote the set of already found edges of H_{un} , i.e., if $e \in E$ and $e \subset F$, then $e \in E'$. Note that a pair (V, E') can be viewed as a partial hypergraph of H_{un} . Let $S, S \subset V$, be a query we deal with. Before we start the proposed algorithm, we set $F = \emptyset$, $E' = \emptyset$ and $S = V$.

Firstly we describe an algorithm depicted as Alg. 2 which allows us to find a new active vertex v , i.e., $v \notin F$. The input of the algorithm are set F and a query S which contain at least one edge $e \in E$ and $e \not\subset E'$. We set $S' = S \setminus F$ and $S'' = S \setminus S'$. At each further step we guarantee that S' contains a new active vertex. While $|S'| > 1$, we run the following procedure. Split up S' into two equal sized subsets S_1 and S_2 , i.e., $S' = S_1 \sqcup S_2$, $|S_1| = \lceil |S'|/2 \rceil$ and $|S_2| = \lfloor |S'|/2 \rfloor$. Then we carry out a query $S_1 \sqcup S''$. If $Q(S_1 \sqcup S'') = 1$ then it means that S_1 contains at least one new active vertex, since from the previous steps of the procedure we have $Q(S'') = 0$. Therefore we set $S' = S_1$ and repeat the procedure. If $Q(S_1 \sqcup S'') = 0$ then at least one new active vertex must lie in S_2 , since from the previous steps we also have $Q(S_1 \sqcup S_2 \sqcup S'') = 1$. Thus we set $S' = S_2$, $S'' = S_1 \sqcup S''$ and repeat the procedure. At final ($|S'| = 1$) we know that the unique vertex v of S' is an active vertex of H_{un} and $v \notin F$. Notice that Alg. 2 can be seen as a variation of the binary vertex search.

Secondly we provide an algorithm depicted as Alg. 3 which allows us to find all edges E' composed on already found active vertices F . The only input of the algorithm is the set F . After we find a new active vertex v we can update the set E' by searching edges containing v . But since $|F| \leq s\ell \ll t$ we can set $E' = \emptyset$ and run the following procedure over all S such that $S \subset F$ and $|S| \leq \ell$. If there is no edge $e \in E'$ such that $e \subset S$, then carry out a query S . If $Q(S) = 1$ then we delete all edges $e \in E'$ such that $S \subset e$ and add edge $e = S$ to E' . Note that Alg. 3 represents an exhaustive search of edges.

Thirdly we present an algorithm depicted as Alg. 4 which allows us to find a query S such that S contains at least one edge $e \in E$ and $e \not\subset E'$ (as a consequence S contains at least one active vertex $v \notin F$). The only input of the algorithm is the set E' . We initialize A as vertices included to at least one edge $e \in E'$, set $B = V \setminus A$ and $S = \emptyset$. One can see that $|A| \leq s\ell \ll t$. Then we run the following procedure for

all sets $C \subset V$ such that $B \subset C$, and $\nexists e \in E', e \subset C$. If $Q(C) = 1$ then we set $S = C$ and exit the procedure. If $Q(C) = 0$ we proceed to the next query C . If we finish the procedure and have $S = \emptyset$, then it means we have found all edges of H_{un} , i.e., $E' = E$. Note that Alg. 4 is an exhaustive query search.

We give a full description of the proposed $\mathcal{F}(t, s, \ell)$ -searching algorithm by Alg. 1, and this algorithm is based on Alg. 2, 3 and 4. We set $F = \emptyset$, $E' = \emptyset$ and $S = V$. While Alg. 3 gives a query $S \neq \emptyset$, we run the following procedure. With the help of Alg. 2 we find a new active vertex v and add it to F . Then we use Alg. 3 to update the set of edges E' composed on already found active vertices F . After that we run Alg. 4 to find an appropriate query S , which then will be used in Alg. 2.

Now we upper bound the number of tests of Alg. 1 in the worst case. Let $|V| = t$. It is easy to check that Alg. 2 uses at most $\lceil \log_2 |S| \rceil \leq \lceil \log_2 t \rceil$ tests. One can see that the number of active vertices in the hidden hypergraph $H_{un} \in \mathcal{F}(t, s, \ell)$ is at most $s\ell$. Alg. 3 uses at most $F_1(s, \ell)$ tests, while Alg. 4 uses at most $F_2(s, \ell)$ tests, where the functions F_1 and F_2 do not depend on t . We can upper bound the number of cycles in Alg. 1 by the number of active vertices. Therefore, the total number of tests for the given algorithm does not exceed $s\ell(\log_2 t + F_1(s, \ell) + F_2(s, \ell) + 1)$. \square

Data: set of vertices V of $H_{un} \in \mathcal{F}(t, s, \ell)$

Result: set of edges E of H_{un}

initialization $E' := \emptyset$; $F := \emptyset$; $S := V$;

while $S \neq \emptyset$ **do**

 perform Alg. 2, find $v \notin F$ and $F := F \sqcup v$;
 perform Alg. 3, and find subset of edges E' ;
 perform Alg. 4, and find query S ;

end

Algorithm 1: Learning a hidden hypergraph

Data: query $S \subseteq V$, $Q(S) = 1$, and set $F \subset V$

Result: vertex $v \in V$, $v \notin F$, and $\exists e \in E$, $v \in e$

initialization $S' := S \setminus F$; $S'' := S \setminus S'$;

while $|S'| > 1$ **do**

 split in half S' : $S' = S_1 \sqcup S_2$;

if $Q(S_1 \sqcup S'') = 1$ **then**

$S' := S_1$;

else

$S' := S_2$, $S'' := S'' \sqcup S_1$;

end

end

Algorithm 2: Searching a new active vertex

IV. CONCEPT OF “ALMOST” LEARNING A HIDDEN HYPERGRAPH

Remind that there are two natural types of algorithms, namely non-adaptive and adaptive. A compromise between these two types is usually called a *multistage* (or *multiple*

Data: set $F \subset V$

Result: subset of edges $E' \subset E$

initialization $E' := \emptyset$;

for $\forall S \subset F$: $1 \leq |S| \leq \ell$ **do**

if $\nexists e \in E' : e \subset S$ **then**

if $Q(S) = 1$ **then**

for $\forall e \in E' : S \subset e$ **do**

 delete e from E' ;

end

 add edge $e = S$ to E' ;

end

end

end

Algorithm 3: Searching edges

Data: subset of edges $E' \subset E$

Result: query S

initialization $A := \{v : v \in e \in E'\}$; $B := V \setminus A$;

$S := \emptyset$;

for $\forall C \subset V$: $B \subset C$ and $\nexists e \in E', e \subset C$ **do**

if $Q(C) = 1$ **then**

$S := C$ and break “for loop”;

end

end

Algorithm 4: Searching a query

round) algorithm. In the given section we will limit ourselves to the consideration of only so called *two-stage searching procedures* (see, e.g., [12]). It means we can adapt the tests of the second round of testing only one time after we receive the answers to the queries of the first round.

Now we consider a relaxation of the problem of learning a hidden hypergraph. Suppose we let an algorithm identify *almost all* hypergraphs from the family $\mathcal{F}(t, s, \ell)$. More formally, if there exist a subfamily $\mathcal{F}'(t, s, \ell) \subset \mathcal{F}(t, s, \ell)$ of cardinality at least $(1 - \varepsilon)|\mathcal{F}(t, s, \ell)|$ such that for any $H_{un} \in \mathcal{F}'(t, s, \ell)$ the algorithm finds H_{un} , then we say that the algorithm is $\mathcal{F}(t, s, \ell)$ -searching with probability $(1 - \varepsilon)$.

By $N^{na}(t, s, \ell, \varepsilon)$ ($N^a(t, s, \ell, \varepsilon)$, $N^{2st}(t, s, \ell, \varepsilon)$) denote the minimal number of queries in a $\mathcal{F}(t, s, \ell)$ -searching non-adaptive (adaptive, two-stage) algorithm with probability $(1 - \varepsilon)$. Introduce the *capacity* of non-adaptive $\mathcal{F}(t, s, \ell)$ -searching algorithms

$$C^{na}(s, \ell) \triangleq \lim_{\substack{\varepsilon \rightarrow 0 \\ t \rightarrow \infty}} \frac{\log_2 t}{N^{na}(t, s, \ell, \varepsilon)}.$$

In similar way we define the capacities $C^a(s, \ell)$ and $C^{2st}(s, \ell)$ of adaptive $\mathcal{F}(t, s, \ell)$ -searching algorithms and two-stage $\mathcal{F}(t, s, \ell)$ -searching algorithms, respectively.

One can easily check that Theorem 1 is true for “almost” concept as well. From definitions it follows

$$C^{na}(s, \ell) \leq C^{2st}(s, \ell) \leq C^a(s, \ell) = \frac{1}{s\ell},$$

where the right-hand side equality holds in virtue of Theorem 2.

For the case $\ell = 1$ the definition of the capacity was considered in many papers. We refer the reader to the classic result [4] in model of designing screening experiments. Malyutov proved that $C^{na}(s, 1) = 1/s$.

We conjecture that $C^{na}(s, \ell) = 1/(s\ell)$. The following theorem reinforces the hypothesis.

Theorem 3. *The capacity of two-stage $\mathcal{F}(t, s, \ell)$ -searching algorithms $C^{2st}(s, \ell) = 1/(s\ell)$.*

We prove Theorem 3 using the probabilistic method and the result established in [4] for the case $\ell = 1$.

Proof of Theorem 3.

Firstly we note that the subfamily $\mathcal{F}'(t, s, \ell) \subset \mathcal{F}(t, s, \ell)$ which consists of s pairwise non-intersecting edges of size ℓ , i.e.,

$$\mathcal{F}'(t, s, \ell) = \left\{ H \in \mathcal{F}(t, s, \ell) : H = (\mathbf{e}_1, \dots, \mathbf{e}_s), \right. \\ \left. |\mathbf{e}_i| = \ell, \mathbf{e}_i \cap \mathbf{e}_j = \emptyset \text{ for any } i \neq j \right\},$$

has cardinality $|\mathcal{F}(t, s, \ell)|(1 + o(1))$ as $t \rightarrow \infty$. Thus, for any $\varepsilon > 0$ and for sufficiently large t it is enough to prove the existence of two-stage $\mathcal{F}'(t, s, \ell)$ -searching algorithm with probability $(1 - \varepsilon)$.

Now we show that there exists a binary matrix (each test corresponds to the row, and each vertex stands for the column) corresponding to the first stage of group testing such that after carrying out tests of the first stage for almost all hypergraphs from $\mathcal{F}'(t, s, \ell)$ we can find a *good* partition of vertices into s disjoint sets: $V_1 \sqcup V_2 \dots \sqcup V_s = V = [t]$, such that $\mathbf{e}_1 \in V_1, \mathbf{e}_2 \in V_2, \dots, \mathbf{e}_s \in V_s$. Define the ensemble $E(N, t, s)$ of s -ary $(N \times t)$ -matrices $X = \|x_i(j)\|$, where each entry $x_i(j)$ is chosen independently and equiprobably from the set $\{1, 2, \dots, s\}$. Each s -ary symbol x in X is then replaced by the binary column of length s , which has only one 1 at x -th position. In other words, the s -ary $(N \times t)$ -matrix X is replaced by the binary $(sN \times t)$ -matrix X_1 , and each s -ary row of X is replaced by the binary $(s \times n)$ -layer of X_1 such that each column of the layer contains only one 1. Define the event $A(s, \ell)$: “given a hypergraph $H \in \mathcal{F}'(t, s, \ell)$, there exists a layer in X_1 such that the answers to all s edge-detecting queries in the layer are 1’s”. If this condition holds, then we are presented with a good partition into disjoint sets: $V_1 \sqcup V_2 \dots \sqcup V_s = V = [t]$, such that $\mathbf{e}_1 \in V_1, \mathbf{e}_2 \in V_2, \dots, \mathbf{e}_s \in V_s$. Denote the cardinality of V_i by $t_i = |V_i|$. Now estimate the probability of the opposite event to $A(s, \ell)$:

$$P(\overline{A(s, \ell)}) = \left(1 - \frac{s!}{s^\ell}\right)^N.$$

Let $N \rightarrow \infty$ and $N = o(\log_2 t)$. Then for any $\varepsilon > 0$ there exists $N(\varepsilon)$ such that for any $N \geq N(\varepsilon)$ the probability $P(\overline{A(s, \ell)}) \leq \varepsilon$. It means that for any $\varepsilon > 0$, and for sufficiently large t and for $N = o(\log_2 t)$ (the number of tests of the first stage is negligible) there exists a binary $(sN \times t)$ -matrix X_1 which can be used in the first stage of the algorithm, and for at least $(1 - \varepsilon)|\mathcal{F}'(t, s, \ell)|$ hypergraphs from $\mathcal{F}'(t, s, \ell)$ there exist s edge-detecting queries, answers of which are 1’s, and supports of these queries are pairwise non-intersecting.

For matrix X we will call all such hypergraphs *good*. For any binary $(N \times t)$ matrix X denote the set of good hypergraphs by $G(X)$, $G(X) \subset \mathcal{F}'(t, s, \ell)$. Notice that we have proved $|G(X_1)| \geq (1 - \varepsilon)|\mathcal{F}'(t, s, \ell)|$.

Now we reformulate the statement derived in [4].

Lemma 1. *The capacity of non-adaptive $\mathcal{F}(t, \ell, 1)$ -searching algorithms $C^{na}(\ell, 1) = 1/\ell$.*

Notice that if we are given with a code X representing a non-adaptive $\mathcal{F}(t, \ell, 1)$ -searching algorithm with probability $(1 - \varepsilon)$ then we can replace each entry of X by the following rule: $0 \rightarrow 1, 1 \rightarrow 0$, and get the code Y , which represents a non-adaptive $\mathcal{F}(t, 1, \ell)$ -searching algorithm with probability $(1 - \varepsilon)$.

Roughly speaking, for any good hypergraph from $G(X_1)$ it will be sufficient to apply s non-adaptive $\mathcal{F}(t_i, 1, \ell)$ -searching algorithms with probabilities $(1 - \varepsilon)$ in parallel at the second stage of our strategy in order to find s edges in each $V_i, t_i = |V_i|$.

More formally, let $E(N, t, X)$ be the ensemble of binary $(N \times t)$ codes that consists of all possible permutations of columns of a code X representing a non-adaptive $\mathcal{F}(t, 1, \ell)$ -searching algorithm with probability $(1 - \varepsilon)$, and each copy of X is chosen equiprobably with probability $1/t!$. Let Y be a random matrix of $E(N, t, X)$. For a good hypergraph $H \in G(X_1)$ let we be given with an appropriate partition into disjoint sets $V_1 \sqcup V_2 \dots \sqcup V_s = V = [t]$ such that $\mathbf{e}_1 \in V_1, \mathbf{e}_2 \in V_2, \dots, \mathbf{e}_s \in V_s$. At the second stage of our strategy for vertices V_1 we will apply N tests of Y without using vertices $V \setminus V_1$, for vertices V_2 we will apply N tests of Y without using vertices $V \setminus V_2$ and so on. Define the event $B(s, \ell)$: “all edges of H can be found applying sN tests”. Estimate the probability of the opposite event to $B(s, \ell)$:

$$P(\overline{B(s, \ell)}) \leq s\varepsilon.$$

It means that there exists such a code X_2 which is a copy (obtained by a column permutation) of X such that for $(1 - s\varepsilon) \cdot |G(X_1)|$ good hypergraphs we find all edges after the second stage of our strategy.

Finally, for any $\varepsilon > 0$ there exists an ascending sequence of t such that for $(1 - \varepsilon)|\mathcal{F}(t, s, \ell)|$ hypergraphs there exists a code X_1 , which can be used for the first stage of $\mathcal{F}(t, s, \ell)$ -searching algorithm with probability $(1 - \varepsilon)$, and a code X_2 , the modification of which can be applied for the second stage of the strategy, such that the total number of tests is sufficiently determined by only queries of the code X_2 and is equal to $s\ell \log_2 t(1 + o(1))$. \square

REFERENCES

- [1] Mitchell C.J., Piper F.C., “Key storage in Secure Networks”, *Discrete Applied Mathematics*, v. 21, pp. 215-228, 1988.
- [2] Dyachkov A. G., Vilenkin P., Macula A., and Torney D., “Families of finite sets in which no intersection of sets is covered by the union of s others”, *J. Combin. Theory. Ser. A*, 99 (2002), pp. 195-218.
- [3] Lebedev V.S. Asymptotic Upper Bound for the Rate of (w,r) Cover-Free Codes // *Problems of Information Transmission*. 2003. V. 39. n 4. P. 317-323.
- [4] Malyutov M.B., “The Separating Property of Random Matrices”, *Mathematical Notes*, vol.23, no. 1, pp. 84-91, 1978.

- [5] *Angluin D., Chen J.*, “Learning a hidden graph using $O(\log n)$ queries per edge”, *J Comput Syst Sci*, v.74, pp. 546-556, 2008.
- [6] *Alon, N., and Asodi, V.*, “Learning a hidden subgraph”. *SIAM J. Discrete Math.* 18, 4 (2005), pp. 697-712.
- [7] *Alon, N., Beigel, R., Kasif, S., Rudich, S., and Sudakov, B.*, “Learning a hidden matching”. *SIAM J. Comput.* 33, 2 (2004), pp. 487-501.
- [8] *Angluin, D., and Chen, J.* “Learning a hidden hypergraph. *Journal of Machine Learning Research* 7 (2006), pp. 2215-2236.
- [9] *Bouvel, M., Grebinski, V., and Kucherov, G.* “Combinatorial search on graphs motivated by bioinformatics applications: A brief survey”. In *WG* (2005), pp. 16-27.
- [10] *Du, D.-Z. and Hwang, F.K.*, “Combinatorial Group Testing and Its Applications”, Singapore: *World Sci.*, 2000, 2nd ed.
- [11] *Abasi, H., Bshouty, N.H., and Mazzawi, H.*, “On Exact Learning Monotone DBF from Membership Queries”, *Lecture Notes in Artificial Intelligence*, 2014, pp. 111-124.
- [12] *De Bonis A., Gasieniec L., Vaccaro U.*, Optimal two-stage algorithms for group testing problems, *SIAM J. Comp.*, vol. 34, no. 5 pp. 1253-1270, 2005.